[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)/[Interchain Developer Academy](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)



Search

[Interchain Developer Academy](https://ida.interchain.io/)[Interchain Developer Academy](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

Search



Filters

Interchain Developer Academy

[](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Week 0 - Getting Started](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Getting Started](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Blockchain 101](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Blockchain History](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Public and Managed Blockchains](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Consensus in Distributed Networks](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Cryptography](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Self-Assessment Quiz](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Go Introduction - First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Go Basics](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Go Interfaces](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Control Structures in Go](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Arrays and Slices in Go](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Standard Packages in Go](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Concurrency in Go](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Good-To-Know Dev Terms](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Docker Introduction](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Week 1 - Introduction to the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Introduction to the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Blockchain Technology and the Interchain](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[The Interchain Ecosystem](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Getting ATOM and Staking It](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[A Blockchain App Architecture](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Accounts](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Messages](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Modules](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Multistore and Keepers](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[BaseApp](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Queries](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Events](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Context](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Testing](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Relaying with IBC](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Interchain Security](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Bridges](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Week 1 Quiz](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Week 2 - First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[First Steps](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Setup Your Work Environment](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Run a Node, API, and CLI](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Ignite CLI](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Exercise - Make a Checkers Blockchain](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Store Object](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Create Custom Messages](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Create and Save a Game Properly](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Add a Way to Make a Move](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Emit Game Information](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Record the Game Winner](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Week 2 Exercise](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Week 3 - Introduction to IBC and CosmJS](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Introduction to IBC and CosmJS](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[What is IBC?](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[IBC/TAO - Connections (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[IBC/TAO - Channels (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[IBC/TAO - Clients (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[IBC Token Transfer](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Interchain Accounts (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[IBC Middleware (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Create a Custom IBC Middleware (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Integrate IBC Middleware Into a Chain (OPTIONAL)](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[IBC Tooling](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[What is CosmJS?](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Your First CosmJS Actions](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Compose Complex Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Learn to Integrate Keplr](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Create Custom CosmJS Interfaces](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Week 4 - Ignite CLI and IBC Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Ignite CLI and IBC Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Keep an Up-To-Date Game Deadline](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Keep Track Of How Many Moves Have Been Played](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Put Your Games in Order](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Auto-Expiring Games](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Let Players Set a Wager](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Handle wager payments](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Integration tests](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Incentivize Players](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Help Find a Correct Move](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Play With Cross-Chain Tokens](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Understand IBC Denoms](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Go Relayer](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Hermes Relayer](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Week 5 - CosmJS Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[CosmJS Advanced](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Create Custom Objects](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Create Custom Messages](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Get an External GUI](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Integrate CosmJS and Keplr](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Backend Script for Game Indexing](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Week 6 - IBC Deep Dive](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[IBC Deep Dive](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[IBC Application Developer Introduction](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Make a Module IBC-Enabled](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Adding Packet and Acknowledgment Data](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Create a Leaderboard Chain](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Week 7 - From Code to MVP to Production and Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[From Code to MVP to Production and Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Run in Production](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Prepare the Software to Run](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Prepare a Validator and Keys](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Prepare Where the Node Starts](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Prepare and Connect to Other Nodes](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Configure, Run, and Set Up a Service](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Prepare and Do Migrations](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Simulate Production in Docker](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Tally Player Info After Production](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Add a Leaderboard as a Module](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Migrate the Leaderboard Module After Production](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Simulate a Migration in Docker](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Final Exam](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[What's Next?](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

[Continue Your Interchain Journey](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html)

Docs Version Switcher

On this page

[Module scope](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#module-scope)

[Module components](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#module-components)

[Interfaces](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#interfaces)

[Protobuf services](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#protobuf-services)

[Msg service](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#msg-service)

[gRPC Query service](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#grpc-query-service)

[Command-line commands](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#command-line-commands)

[Keeper](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#keeper)

[Core modules](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#core-modules)

[Recommended folder structure](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#recommended-folder-structure)

[Structure](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#structure)

[Errors](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#errors)

[Registration](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#registration)

[Wrapping](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#wrapping)

[ABCI](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#abci)

[Code example](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#code-example)

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#modules) **Modules**



Review the following sections to better understand modules in the Cosmos SDK:

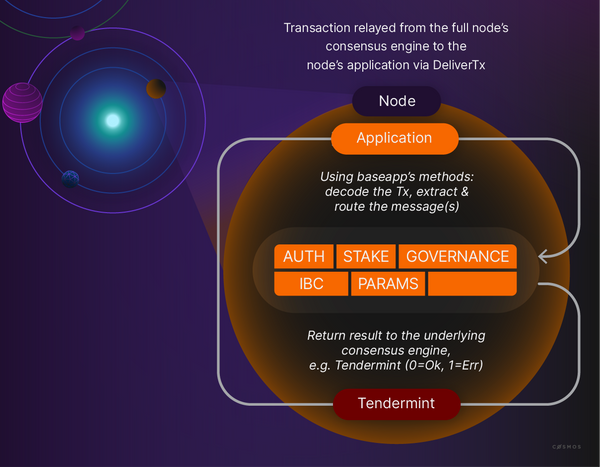
* [Transactions](https://ida.interchain.io/academy/2-cosmos-concepts/3-transactions.html)
* [Messages](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)
* [Queries](https://ida.interchain.io/academy/2-cosmos-concepts/9-queries.html)



Modules are functional components that address application-level concerns such as token management or governance. The Cosmos SDK includes several ready-made modules so that application developers can focus on the truly unique aspects of their application.   
  
A code example that illustrates module creation and an introduction to your checkers blockchain can be found at the end of this section.

Each Cosmos chain is a purpose-built blockchain. Cosmos SDK modules define the unique properties of each chain. Modules can be considered state machines within the larger state machine. They contain the storage layout, or state, and the state transition functions, which are the message methods.

Modules define most of the logic of Cosmos SDK applications.



When a transaction is relayed from the underlying CometBFT consensus engine, BaseApp decomposes the Messages contained within the transaction and routes messages to the appropriate module for processing. Interpretation and execution occur when the appropriate module message handler receives the message.

Developers compose together modules using the Cosmos SDK to build custom application-specific blockchains.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#module-scope) Module scope

Modules include **core** functionality that every blockchain node needs:

* A boilerplate implementation of the Application Blockchain Interface (ABCI) that communicates with CometBFT.
* A general-purpose data store that persists the module state called multistore.
* A server and interfaces to facilitate interactions with the node.

Modules implement the majority of the application logic while the **core** attends to wiring and infrastructure concerns, and enables modules to be composed into higher-order modules.

A module defines a subset of the overall state, using:

* One or more keys or value stores, known as KVStore.
* A subset of message types that are needed by the application and do not exist yet.

Modules also define interactions with other modules that already exist.



Most of the work for developers involved in building a Cosmos SDK application consists of building custom modules required by their application that do not exist yet, and integrating them with modules that already exist into one coherent application. Existing modules can come either from the Cosmos SDK itself or from **third-party developers**. You can download these from an online module repository.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#module-components) Module components

It is best practice to define a module in the x/moduleName folder. For example, the module called Checkers would go in x/checkers. If you look at the Cosmos SDK's base code, it also [defines its modules (opens new window)↗](https://github.com/cosmos/cosmos-sdk/tree/main/x) in an x/ folder.

Modules implement several elements:

****

**Interfaces**

Interfaces facilitate communication between modules and the composition of multiple modules into coherent applications.

****

**Protobuf**

Protobuf provides one Msg service to handle messages and one gRPC Query service to handle queries.

****

**Keeper**

A Keeper is a controller that defines the state and presents methods for updating and inspecting the state.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#interfaces) Interfaces

A module must implement **three application module interfaces** to be integrated with the rest of the application:

* **AppModuleBasic:** implements non-dependent elements of the module.
* **AppModule:** interdependent, specialized elements of the module that are unique to the application.
* **AppModuleGenesis:** interdependent, genesis/initialization elements of the module that establish the initial state of the blockchain at inception.

You define AppModule and AppModuleBasic, and their functions, in your module's x/moduleName/module.go file.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#protobuf-services) Protobuf services

Each module defines two Protobuf services:

* **Msg:** a set of RPC methods related one-to-one to Protobuf request types, to handle messages.
* **Query:** gRPC query service, to handle queries.



If this topic is new to you, read an introduction to [Protocol Buffers (opens new window)↗](https://www.ionos.com/digitalguide/websites/web-development/protocol-buffers-explained/).

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#msg-service) Msg service

Regarding the Msg service, keep in mind:

* Best practice is to define the Msg Protobuf service in the tx.proto file.
* Each module should implement the RegisterServices method as part of the AppModule interface. This lets the application know which messages and queries the module can handle.
* Service methods should use a *keeper*, which encapsulates knowledge about the storage layout and presents methods for updating the state.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#grpc-query-service) gRPC Query service

For the gRPC Query service, keep in mind:

* Best practice is to define the Query Protobuf service in the query.proto file.
* It allows users to query the state using gRPC.
* Each gRPC endpoint corresponds to a service method, named with the rpc prefix inside the gRPC Query service.
* It can be configured under the grpc.enable and grpc.address fields in app.toml.

Protobuf generates a QueryServer interface containing all the service methods for each module. Modules implement this QueryServer interface by providing the concrete implementation of each service method in separate files. These implementation methods are the handlers of the corresponding gRPC query endpoints. This division of concerns across different files makes the setup safe from a re-generation of files by Protobuf.



[gRPC (opens new window)↗](https://grpc.io/) is a modern, open-source, high-performance framework that supports multiple languages. It is the recommended standard for external clients such as wallets, browsers, and backend services to interact with a node.

gRPC-Gateway REST endpoints support external clients that may not wish to use gRPC. The Cosmos SDK provides a gRPC-gateway REST endpoint for each gRPC service.



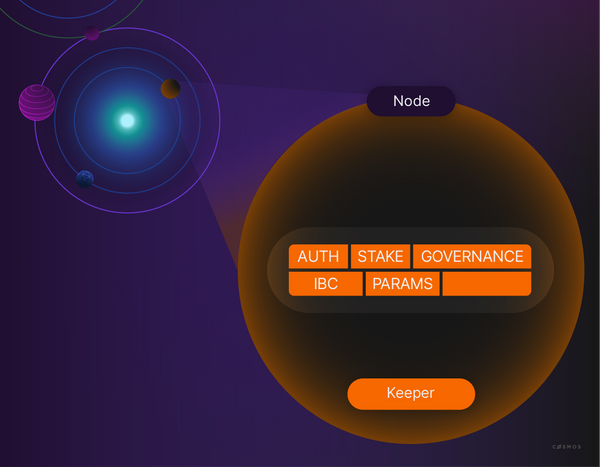
See the [gRPC-Gateway documentation (opens new window)↗](https://grpc-ecosystem.github.io/grpc-gateway/) for more on the gRPC-Gateway plugin.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#command-line-commands) Command-line commands

Each module defines commands for a command-line interface (CLI). Commands related to a module are defined in a folder called client/cli. The CLI divides commands into two categories: transactions and queries. These are the same as those which you defined in tx.go and query.go respectively.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#keeper) Keeper

Keepers are the gatekeepers to any stores in the module. It is mandatory to go through a module’s keeper to access a store. A keeper encapsulates the knowledge about the layout of the storage within the store and contains methods to update and inspect it. If you come from a module-view-controller (MVC) world, then it helps to think of the keeper as the controller.



Other modules may need access to a store, but other modules are also potentially malicious or poorly written. For this reason, developers need to consider who and what should have access to their module stores. To prevent a module from randomly accessing another module at runtime, a module needs to declare its intent to use another module at construction. At this point, such a module is granted a runtime key that lets it access the other module. Only modules that hold this key to a store can access the store. This is part of what is called an **object-capability model**.

Keepers are defined in keeper.go. A keeper's type definition generally consists of keys to the module's own store in the multistore, references to the keepers of other modules, and a reference to the application's codec.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#core-modules) Core modules

The Cosmos SDK includes a set of core modules that address common concerns with well-solved, standardized implementations. Core modules address application needs such as tokens, staking, and governance.

Core modules offer several advantages over ad-hoc solutions:

* Standardization is established early, which helps ensure good interoperability with wallets, analytics, other modules, and other Cosmos SDK applications.
* Duplication of effort is significantly reduced because application developers focus on what is unique about their application.
* Core modules are working examples of Cosmos SDK modules that provide strong hints about suggested structure, style, and best practices.

Developers create coherent applications by selecting and composing core modules first and then implementing the custom logic.



Why not explore the [list of core modules and the application concerns they address (opens new window)↗](https://github.com/cosmos/cosmos-sdk/tree/main/x)?

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#recommended-folder-structure) Recommended folder structure



The following ideas are meant to be applied as suggestions. Application developers are encouraged to improve and contribute to the module structure and development design.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#structure) Structure

A typical Cosmos SDK module can be structured as follows:

1. The serializable data types and Protobuf interfaces:



Copy

proto

└── {project\_name}

└── {module\_name}

└── {proto\_version}

├── {module\_name}.proto

├── event.proto

├── genesis.proto

├── query.proto

└── tx.proto

* {module\_name}.proto: the module's common message type definitions.
* event.proto: the module's message type definitions related to events.
* genesis.proto: the module's message type definitions related to the genesis state.
* query.proto: the module's Query service and related message type definitions.
* tx.proto: the module's Msg service and related message type definitions.

1. Then the rest of the code elements:



Copy

x/{module\_name}

├── client

│ ├── cli

│ │ ├── query.go

│ │ └── tx.go

│ └── testutil

│ ├── cli\_test.go

│ └── suite.go

├── exported

│ └── exported.go

├── keeper

│ ├── genesis.go

│ ├── grpc\_query.go

│ ├── hooks.go

│ ├── invariants.go

│ ├── keeper.go

│ ├── keys.go

│ ├── msg\_server.go

│ └── querier.go

├── module

│ └── module.go

├── simulation

│ ├── decoder.go

│ ├── genesis.go

│ ├── operations.go

│ └── params.go

├── spec

│ ├── 01\_concepts.md

│ ├── 02\_state.md

│ ├── 03\_messages.md

│ └── 04\_events.md

├── {module\_name}.pb.go

├── abci.go

├── codec.go

├── errors.go

├── events.go

├── events.pb.go

├── expected\_keepers.go

├── genesis.go

├── genesis.pb.go

├── keys.go

├── msgs.go

├── params.go

├── query.pb.go

└── tx.pb.go

* client/: the module's CLI client functionality implementation and the module's integration testing suite.
* exported/: the module's exported types - typically interface types (see also the following note).
* keeper/: the module's Keeper and MsgServer implementations.
* module/: the module's AppModule and AppModuleBasic implementations.
* simulation/: the module's simulation package defines functions used by the blockchain simulator application (simapp).
* spec/: the module's specification documents outlining important concepts, state storage structure, and message and event type definitions.
* The root directory includes type definitions for messages, events, and genesis state, including the type definitions generated by Protocol Buffers:
  + abci.go: the module's BeginBlocker and EndBlocker implementations. This file is only required if BeginBlocker and/or EndBlocker need to be defined.
  + codec.go: the module's registry methods for interface types.
  + errors.go: the module's sentinel errors.
  + events.go: the module's event types and constructors.
  + expected\_keepers.go: the module's expected other keeper interfaces.
  + genesis.go: the module's genesis state methods and helper functions.
  + keys.go: the module's store keys and associated helper functions.
  + msgs.go: the module's message type definitions and associated methods.
  + params.go: the module's parameter type definitions and associated methods.
  + \*.pb.go: the module's type definitions generated by Protocol Buffers as defined in the respective \*.proto files.



If a module relies on keepers from another module, the exported/ code element expects to receive the keepers as interface contracts to avoid a direct dependency on the module implementing the keepers. However, these interface contracts can define methods that operate on (or return types that are specific to) the module that is implementing the keepers.   
  
The interface types defined in exported/ use canonical types that allow for the module to receive the interface contracts through the expected\_keepers.go file. This pattern allows for code to remain [DRY (opens new window)↗](https://en.wikipedia.org/wiki/Don%27t_repeat_yourself) and also alleviates import cycle chaos.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#errors) Errors

Modules are encouraged to define and register their own errors to provide better context for failed messages or handler executions. Errors should be common or general errors, which can be further wrapped to provide additional specific execution context.



For more details see the [Cosmos SDK documentation on errors when building modules (opens new window)↗](https://docs.cosmos.network/main/building-modules/errors.html).

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#registration) Registration

Modules should define and register their custom errors in x/{module}/errors.go. Registration of errors is handled via the types/errors package.

Each custom module error must provide the codespace, which is typically the module name (for example, "distribution") and is unique per module, and a uint32 code. The codespace and code together provide a globally unique Cosmos SDK error.

The only restrictions on error codes are the following:

* They must be greater than one, as a code value of one is reserved for internal errors.
* They must be unique within the module.



The Cosmos SDK provides a core set of common errors. These errors are defined in [types/errors/errors.go (opens new window)↗](https://github.com/cosmos/cosmos-sdk/blob/master/types/errors/errors.go).

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#wrapping) Wrapping

The custom module errors can be returned as their concrete type, as they already fulfill the error interface. Module errors can be wrapped to provide further context and meaning to failed executions.

Regardless of whether an error is wrapped or not, the Cosmos SDK's errors package provides an API to determine if an error is of a particular kind via Is.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#abci) ABCI

If a module error is registered, the Cosmos SDK errors package allows ABCI information to be extracted through the ABCIInfo API. The package also provides ResponseCheckTx and ResponseDeliverTx as auxiliary APIs to automatically get CheckTx and DeliverTx responses from an error.

[#Copy link](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#code-example) Code example

****

**Show me some code for my checkers blockchain**

Now your application is starting to take shape. **The checkers module**

When you create your checkers blockchain application, you ought to include a majority of the standard modules like auth, bank, and so on. With the Cosmos SDK boilerplate in place, the *checkers part* of your checkers application will most likely reside in a single checkers module. This is the module that you author. **Game wager**

Earlier the goal was to let players play with *money*. With the introduction of modules like bank you can start handling that. The initial ideas are:

* The wager amount is declared when creating a game.
* Each player is billed the amount when making their first move, which is interpreted as "challenge accepted". The amount should not be deducted on the game creation. If the game times out, the first player gets refunded.
* Subsequent moves by a player do not cost anything.
* If a game ends in a win or times out on a forfeit, the winning player gets the total wager amount.
* If a game ends in a draw, then both players get back their amount.

How would this look in terms of code? You need to add the wager to:

* The game:

Copy

type StoredGame struct {

...

Wager uint64

}

* The message to create a game:

Copy

type MsgCreateGame struct {

...

Wager uint64

}

**Wager payment**

Now you must decide how the tokens are moved. When a player accepts a challenge, the amount is deducted from that player's balance. But where does it go? You could burn the tokens and re-mint them at a later date, but this would make the total supply fluctuate wildly for no apparent benefit. It is possible to transfer from a player to a module. The module acts as the escrow account for all games. So when playing for the first time, a player would:

Copy

import (

sdk "github.com/cosmos/cosmos-sdk/types"

bankKeeper "github.com/cosmos/cosmos-sdk/x/bank/keeper"

)

wager := sdk.NewCoin("stake", sdk.NewInt(storedGame.Wager))

payment := sdk.NewCoins(wager)

var bank bankKeeper.Keeper

var ctx sdk.Context

var playerAddress sdk.AccAddress

err := bank.SendCoinsFromAccountToModule(ctx, playerAddress, "checkers", payment)

if err != nil {

return errors.New("player cannot pay the wager")

}

"stake" identifies the likely name of the base token of your application, the token that is used with the consensus. Of course, you can choose to have the wager paid in any denomination, even one decided by players when creating the game. Conversely, when paying a winner you would have:

Copy

amount := sdk.NewInt(storedGame.Wager).Mul(sdk.NewInt(2))

winnings := sdk.NewCoin("stake", amount)

payment := sdk.NewCoins(winnings)

var winnerAddress sdk.AccAddress

err := bank.SendCoinsFromModuleToAccount(ctx, "checkers", winnerAddress, payment)

if err != nil {

panic("Cannot pay the winnings to winner")

}

Note that:

* It is a *standard* error when the player cannot pay, which is *easily* fixed by the player.
* It is a panic (an internal error) when the escrow account cannot pay, because if the escrow cannot pay it means there is a logic problem somewhere.



If you want to go beyond the code samples in the expandable above and instead see in more detail how to define all this, go to [Run Your Own Cosmos Chain](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/).

More specifically, you can jump to:

* [Ignite CLI](https://ida.interchain.io/hands-on-exercise/1-ignite-cli/1-ignitecli.html) to create a new blockchain with your checkers module.
* The advanced [Handle Wager Payments](https://ida.interchain.io/hands-on-exercise/2-ignite-cli-adv/6-payment-winning.html) to have your checkers module access and use the bank module.
* [Add a leaderboard module](https://ida.interchain.io/hands-on-exercise/4-run-in-prod/3-add-leaderboard.html) and loosely couple it with another module with the use of the hooks pattern. It also leverages the module's Params construct.
* The IBC-advanced [Extend the Checkers Game With a Leaderboard](https://ida.interchain.io/hands-on-exercise/5-ibc-adv/6-ibc-app-checkers.html) to add a second custom IBC module to your checkers blockchain.
* The IBC-advanced [Create a Leaderboard Chain](https://ida.interchain.io/hands-on-exercise/5-ibc-adv/7-ibc-app-leaderboard.html) to create a new blockchain with a module, once again.

synopsis

To summarize, this section has explored:

* How Cosmos SDK modules can be viewed as purpose-specific state machines that define the unique properties of the larger state machine that is each blockchain.
* How messages are decomposed from the incoming transaction containing them and routed to the appropriate module for processing.
* How all modules comprise three core functionalities: an implementation of ABCI to communicate with CometBFT; a general-purpose data store which persists the module state; and the server and interfaces which facilitate interactions with the node.
* How the majority of work for developers is in building custom modules that satisfy their unique needs, which are then integrated into a coherent application alongside existing modules from the Cosmos SDK of third-party developers.
* How the Cosmos SDK's set of core modules address common applications needs (such as tokens, staking, and governance) while providing useful benefits like standardization across the Ecosystem, less duplication of effort, and practical examples of effective structure, style, and best practices.
* How modules should ideally define and register their own set of errors (in addition to the Cosmos SDK's set of common errors), allowing developers to add context and meaning to failed executions.

previous

[](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)

**[Messages](https://ida.interchain.io/academy/2-cosmos-concepts/4-messages.html)**

up next

**[Protobuf](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html)**

[[](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html)](https://ida.interchain.io/academy/2-cosmos-concepts/6-protobuf.html)

Rate this Page

icon smile

icon meh

icon frown

Would you like to add a message?

Submit

Thank you for your Feedback!

[](https://ida.interchain.io/ida-course/discord-info.html)

On this page

[Module scope](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#module-scope)

[Module components](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#module-components)

[Interfaces](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#interfaces)

[Protobuf services](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#protobuf-services)

[Msg service](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#msg-service)

[gRPC Query service](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#grpc-query-service)

[Command-line commands](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#command-line-commands)

[Keeper](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#keeper)

[Core modules](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#core-modules)

[Recommended folder structure](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#recommended-folder-structure)

[Structure](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#structure)

[Errors](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#errors)

[Registration](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#registration)

[Wrapping](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#wrapping)

[ABCI](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#abci)

[Code example](https://ida.interchain.io/academy/2-cosmos-concepts/5-modules.html#code-example)

#### **Get Cosmos updates**

Unsubscribe at any time. [Privacy Policy↗](https://v1.cosmos.network/privacy)

     Next

Documentation

[Cosmos SDK](https://docs.cosmos.network/)[Cosmos Hub](https://hub.cosmos.network/)[CometBFT](https://docs.cometbft.com/)[IBC Protocol](https://ibc.cosmos.network/)

Community

[Interchain blog](https://blog.cosmos.network/)[Forum](https://forum.cosmos.network/)[Discord](https://discord.gg/cosmosnetwork)

Contributing

[Source code on GitHub](https://github.com/cosmos/sdk-tutorials)

[](https://ida.interchain.io/)

[Interchain Developer Academy](https://ida.interchain.io/)

**[](https://blog.cosmos.network/)[](https://twitter.com/cosmos)[](https://discord.gg/cosmosnetwork)[](https://www.linkedin.com/company/interchain-foundation/about/)[](https://reddit.com/r/cosmosnetwork)[](https://t.me/cosmosproject)[](https://www.youtube.com/c/CosmosProject)**



Dark mode

† This website is maintained by the Interchain Foundation (ICF). The contents and opinions of this website are those of the ICF. The ICF provides links to cryptocurrency exchanges as a service to the public. The ICF does not warrant that the information provided by these websites is correct, complete, and up-to-date. The ICF is not responsible for their content and expressly rejects any liability for damages of any kind resulting from the use, reference to, or reliance on any information contained within these websites.

Cosmos is a registered trademark of the [Interchain Foundation.](https://interchain.io/)[Privacy](https://v1.cosmos.network/privacy)